Q1

a) $softmax(x) = softmax(x + c)$

$$softmax(x+c) = \frac{e^{x+c}}{\sum_{k=1}^{K} e^{x_k+c}} = \frac{e^{x}e^{c}}{\sum_{k=1}^{K} e^{x_k}e^{c}} = \frac{e^{x}}{\sum_{k=1}^{K} e^{x_k}} = softmax(x)$$

Q2

a) $\sigma(x) = \frac{1}{1+e^{-x}}$

$$\frac{\partial\sigma(x)}{\partial x} = \frac{(1+e^{-x})(0)-(1)(-e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1+e^{-x}-1}{(1+e^{-x})^2} = \left[\frac{1}{1+e^{-x}}\right]\left[\frac{1+e^{-x}-1}{1+e^{-x}}\right] =$$

$$\sigma(x)[1-\sigma(x)]$$

b) $CE(y,\hat{y}) = -\sum_{k=1}^{K} y_k log(\hat{y}_k)$

$$If\ i = j:\ \frac{\partial\hat{y}_i}{\partial x_j} = \frac{(\sum_{k=1}^{K} e^{x_k})(e^{x_j}) - (e^{x_j})(e^{x_j})}{(\sum_{k=1}^{K} e^{x_k})^2} = (1)(\hat{y}_j) - (\hat{y}_j)^2$$

$$= (\hat{y}_j)(1-\hat{y}_j)$$

$$If\ i \neq j:\frac{\partial\hat{y}_i}{\partial x_j} = \frac{(\sum_{k=1}^{K} e^{x_k})(0) - (e^{x_i})(e^{x_j})}{(\sum_{k=1}^{K} e^{x_k})^2} = -\hat{y}_i\hat{y}_j$$

$$\frac{\partial CE(y,\hat{y})}{\partial x_j} = -\left[\sum_{k\neq j}^{K}\frac{y_k}{\hat{y}_k}(-\hat{y}_k\hat{y}_j) + \frac{y_j}{\hat{y}_j}(\hat{y}_j)(1-\hat{y}_j)\right]$$

$$= -\left[\sum_{k\neq j}^{K} -y_k\hat{y}_j + (y_j)(1-\hat{y}_j)\right]$$

$$= -\left[\sum_{k\neq j}^{K}(-y_k\hat{y}_j - y_j\hat{y}_j) + (y_j)\right]$$

$$= -\left[(-\hat{y}_j)\left[\sum_{k\neq j}^{K}(y_k) + y_j\right] + (y_j)\right]$$

$$= -\left[(-\hat{y}_j)\left[\sum_{k=1}^{K}(y_k)\right] + (y_j)\right] = -[(-\hat{y}_j)(1) + (y_j)]$$

$$= \hat{y}_j - y_j$$

c) $h = \sigma(XW_1 + b_1) \qquad \hat{y} = softmax(hW_2 + b_2) \quad J = CE(y, \hat{y})$

$z = hW_2 + b_2 \quad a = XW_1 + b_1$

$$\frac{\partial J}{\partial X} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial h}\frac{\partial h}{\partial a}\frac{\partial a}{\partial X} = (\hat{y} - y)(W_2)\big(\sigma'(a)\big)(W_1)$$
$$= (\hat{y} - y)(W_2)^T \circ \sigma(XW_1 + b_1)(1 - \sigma(XW_1 + b_1))(W_1)^T$$

d) Number of parameters = $\left(H \times D_y\right) + D_y + (D_x \times H) + H$

Q3

a) Softmax Skip-gram

$$\hat{y}_o = p(o|c) = \frac{e^{(u_o{}^T v_c)}}{\sum_{k=1}^{V} e^{(u_k{}^T v_c)}}$$
$$J_{softmax-CE}(o, U, v_c) = CE(y, \hat{y})$$
$$U = [u_1, \dots, u_v]$$

$$\frac{\partial J}{\partial v_c} = \frac{\partial\left[-\sum_{j=1}^{V} y_j \log\left(\frac{e^{(u_j{}^T v_c)}}{\sum_{k=1}^{V} e^{(u_k{}^T v_c)}}\right)\right]}{\partial v_c}$$
$$= \sum_{j=1}^{V} y_j \left[-\log e^{(u_j{}^T v_c)} + \log \sum_{k=1}^{V} e^{(u_k{}^T v_c)}\right] \Big/ \partial v_c$$
$$= \sum_{j=1}^{V} y_j \left[-u_j{}^T v_c + \log \sum_{k=1}^{V} e^{(u_k{}^T v_c)}\right] \Big/ \partial v_c$$
$$= y_o \left[-u_o{}^T v_c + \log \sum_{k=1}^{V} e^{(u_k{}^T v_c)}\right] \Big/ \partial v_c$$
$$= (1)\left[-u_o{}^T v_c + \log \sum_{k=1}^{V} e^{(u_k{}^T v_c)}\right] \Big/ \partial v_c$$
$$= -u_o{}^T v_c + \log \sum_{k=1}^{V} e^{(u_k{}^T v_c)} \Big/ \partial v_c = -u_o + \sum_{i=1}^{V} \frac{u_i e^{(u_i{}^T v_c)}}{\sum_{k=1}^{V} e^{(u_k{}^T v_c)}}$$
$$= -u_o + \sum_{i=1}^{V} u_i \hat{y}_i = U^T \hat{y} - U^T y = U^T[\hat{y} - y]$$

b) $\frac{\partial J}{\partial u_k} = \frac{\partial\left[-\sum_{j=1}^{V} y_j \log\left(\frac{e^{(u_j{}^T v_c)}}{\sum_{i=1}^{V} e^{(u_i{}^T v_c)}}\right)\right]}{\partial u_k} =$

$$-\left[\sum_{j\neq k} y_j\left[-log\sum_{i=1}^{V} e^{(u_i{}^T v_c)}\right] + y_k\left[\log e^{(u_j{}^T v_c)} - log\sum_{i=1}^{V} e^{(u_i{}^T v_c)}\right]\right] \Big/ \partial u_k$$

$$= \sum_{j\neq k} y_j \frac{v_c e^{(u_k{}^T v_c)}}{\sum_{i=1}^{V} e^{(u_i{}^T v_c)}} - y_k\left[v_c - \frac{v_c e^{(u_k{}^T v_c)}}{\sum_{i=1}^{V} e^{(u_i{}^T v_c)}}\right]$$

$$= \sum_{j=1}^{K} y_j \frac{v_c e^{(u_k{}^T v_c)}}{\sum_{i=1}^{V} e^{(u_i{}^T v_c)}} - y_k v_c = (1)\frac{v_c e^{(u_k{}^T v_c)}}{\sum_{i=1}^{V} e^{(u_i{}^T v_c)}} - y_k v_c$$

$$= v_c\hat{y}_k - v_c y_k = v_c(\hat{y}_k - y_k)$$

$$\frac{\partial J}{\partial U} = (\hat{y} - y)v_c{}^T$$

c) Negative Sampling Skip-gram

$$J_{neg-sample}(o, U, v_c) = -\log\sigma(u_o{}^T v_c) - \sum_{k=1}^{K}\log\sigma(-u_k{}^T v_c)$$

$$\frac{\partial J}{\partial v_c} = -u_o\frac{\sigma'^{(u_o{}^T v_c)}}{\sigma(u_o{}^T v_c)} + \sum_{k=1}^{K} u_k\frac{\sigma'^{(-u_k{}^T v_c)}}{\sigma(-u_k{}^T v_c)}$$

$$= -u_o\frac{\sigma(u_o{}^T v_c)\big(1-\sigma(u_o{}^T v_c)\big)}{\sigma(u_o{}^T v_c)}$$

$$+ \sum_{k=1}^{K} u_k\frac{\sigma(-u_k{}^T v_c)\big(1-\sigma(-u_k{}^T v_c)\big)}{\sigma(-u_k{}^T v_c)}$$

$$= u_o(\sigma(u_o{}^T v_c) - 1) + \sum_{k=1}^{K} u_k\big(1-\sigma(-u_k{}^T v_c)\big)$$

$$= u_o(\sigma(u_o{}^T v_c) - 1) + \sum_{k=1}^{K} u_k\big(\sigma(u_k{}^T v_c)\big)$$

$$\frac{\partial J}{\partial u_j} = v_c(\sigma(u_o{}^T v_c) - 1) + \sum_{k=1}^{K} v_c\big(\sigma(u_k{}^T v_c)\big)$$

$$If\ j = o, \frac{\partial J}{\partial u_j} = v_c\left(\sigma\left({u_j}^T v_c\right) - 1\right)$$

$$If\ j \neq o\ and\ j \in\ K, \frac{\partial J}{\partial u_j} = v_c\left(\sigma\left({u_j}^T v_c\right)\right)$$

$$If\ j \neq o\ and\ j \notin\ K, \frac{\partial J}{\partial u_j} = 0$$

Use a for-loop to go through each negative sample, since negative samples could be duplicates. Just slicing from U-matrix will not update duplicate vectors multiple times.

Negative-sampling does not need to compute $e^{({u_i}^T v_c)}$ for every word in the vocabulary for normalization, hence it is much faster as it only does this for K negative samples.

d) $J_{skip-gram} = \sum_{-m \le j \le m, j \neq 0} F(w_{t+j}, v_c)$

$$\frac{\partial J}{\partial v_c} = \sum_{-m \le j \le m, j \neq 0} \frac{\partial F\left(w_{t+j}, v_c\right)}{\partial v_c}$$

$$\frac{\partial J}{\partial v_{w_{t+j}}} = \mathbf{0}\ for\ all\ j \neq 0\ and\ w_t = c$$

$$\frac{\partial J}{\partial u_k} = \sum_{-m \le j \le m, j \neq 0} \frac{\partial F\left(w_{t+j}, v_c\right)}{\partial u_k}, where\ w_{t+j} = k\ and\ k\ is\ arbitrary$$

$$\frac{\partial J}{\partial U} = \sum_{-m \le j \le m, j \neq 0} \frac{\partial F\left(w_{t+j}, v_c\right)}{\partial U}$$

$J_{CBOW} = F(w_t, \hat{\boldsymbol{v}}),\ \hat{\boldsymbol{v}} = \sum_{-m \le j \le m, j \neq 0} v_{w_{t+j}}$

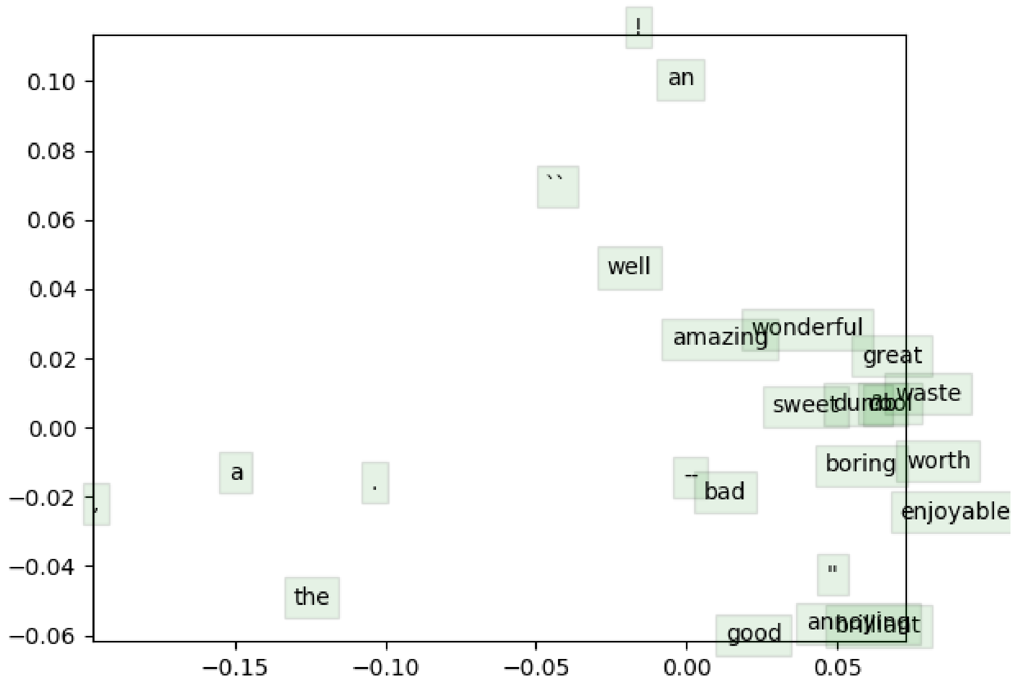
$$\frac{\partial J}{\partial v_{w_{t+j}}} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}} \frac{\partial \hat{\boldsymbol{v}}}{\partial v_{w_{t+j}}} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}} \circ \mathbf{1}_{D_v} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}$$

$$for\ all\ j\ \in \{-m, \dots, -1, +1, \dots, +m\}$$

$$\frac{\partial J}{\partial v_{w_{t+j}}} = \mathbf{0}\ for\ all\ j\ \in \{-m, \dots, -1, +1, \dots, +m\}$$

$$\frac{\partial J}{\partial u_k} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial u_k}; \ \frac{\partial J}{\partial U} = \frac{\partial F(w_t, \hat{\boldsymbol{v}})}{\partial U}$$

g)



Distance between word vectors representing adjectives are short. Common word vectors like “a”, “the”, “,” and “.” are also grouped closer together. Certain words that do not have close meaning are similar in vector space such as “brilliant” with “annoying”.

Q4

b) Regularization prevents overfitting by ensuring the weights are not too large, hence the model generalizes well to unseen test data.

c) `bestResult = max(results, key = lambda x: x[‘dev’])`

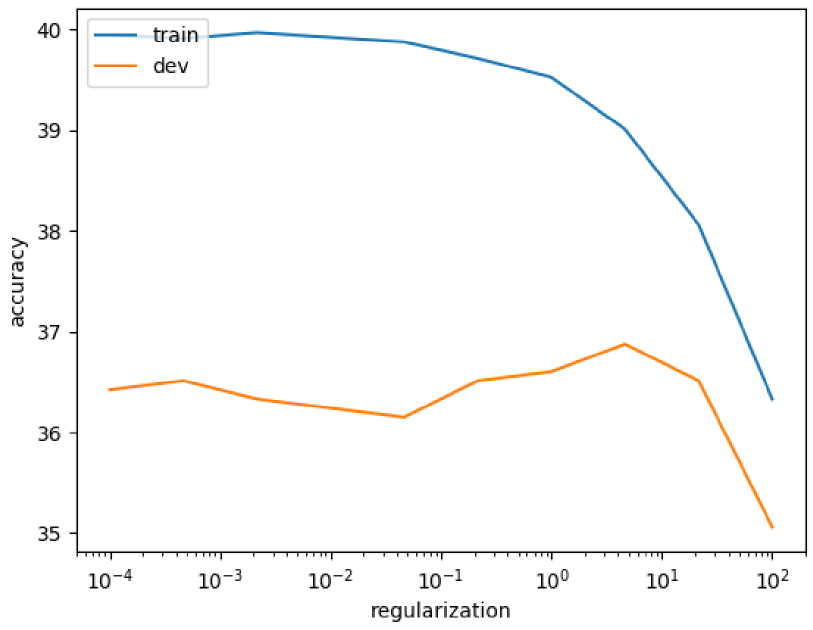
d)

	Yourvectors	pretrained
Regularization Strength	1.00E-04	4.64
Train Accuracy	31.063	39.010
Dev Accuracy	32.516	36.876
Test Accuracy	30.361991	37.285

- Pretrained vectors did better since they were trained on a large amount of data;
- The word vectors are higher-dimensional so they can capture the underlying features of the words better;
- GloVe captures not only the similarity of word vectors, but also the co-occurrence of words in the form of log probabilities, making the model more complex but also

better able to capture the semantic information compared to word2vec, which uses only the dot product, a scalar value that is too simple and crude.

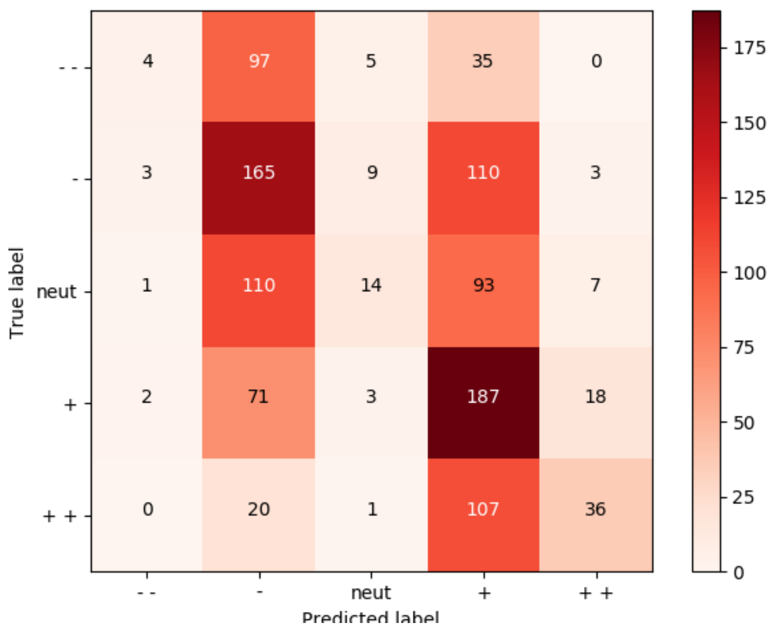
e)



If the regularization strength is too large, the model will not perform well for both training and test set data as the weights are too small. The data dominates the accuracy of the model, hence it can be said to underfit.

If regularization strength is too small, then the model overfits since the weights are large and they dominate the model accuracy. Since the weights are tuned during training only, it means the model places a higher priority on what it has learned during training.

f)



True label \ Predicted label	- -	-	neut	+	+ +
- -	4	97	5	35	0
-	3	165	9	110	3
neut	1	110	14	93	7
+	2	71	3	187	18
+ +	0	20	1	107	36

The model predicts well for + and - labels. It makes the most mistakes when predicting + and - but the true label is – and neutral respectively. The model also mostly tries to predict either + or – labels.

g) 3 4 it 's a lovely film with lovely performances by buy and accorsi .
The model predicts ++ but the true label is +.

2 1 no one goes unindicted here , which is probably for the best .
The model predicts - but the true label is neutral. The model likely considered the word “”no” as hugely negative, so it predicts a -, but in fact the actual meaning lies in the phrase “no one” instead of “no”. Also, averaging word vectors does not allow negation, so “probably for the best”, which is a good-sentiment phrase, did not help to raise the overall sentiment from negative to neutral.

3 1 and if you 're not nearly moved to tears by a couple of scenes , you 've got ice water in your veins .
The model predicts - but the true label is +. The model fails to understand that the movie is actually good since it’s touching, it sees words like “not”, “tears”, and perhaps “ice”, it believes the sentiment is bad. The model needs to learn to understand “moved to tears” is good sentiment in this context.